An Unsplit Godunov Method for Systems of Conservation Laws on Curvilinear Overlapping Grids

David L. Brown [1]

July 1993
Los Alamos Unclassified Report LA-UR-93-2868
to appear in
Mathematical and Computer Modelling

[1]Computing and Communications Division Los Alamos National Laboratory , Los Alamos, New Mexico, USA; this work performed under the auspices of the U.S. Department of Energy by Los Alamos National Laboratory under Contract W-7405-ENG-36.

Contents

1 Introduction **1**

2 The Difference Scheme **2**
2.1 Notation and Grid Setup 4
2.2 The corrector step 5
2.3 The predictor step 5
2.4 The approximate flux function 10
2.5 Alternative Finite-Volume differencing 11
2.6 Overlapping grids 12
2.7 Updating the Solution on an Overlapping Grid 13

3 The Euler Equations **14**

4 Boundary Conditions for Euler Flow **18**
4.1 Solid Walls 20
4.2 Far-field and Outflow Boundary Conditions 22
4.3 Corner Boundary Conditions 22

5 Computational examples **23**

Abstract

The overlapping grid or Chimera method for representation of complex geometries within the context of finite difference methods has been used effectively for a variety of problems, particularly in the area hydrodynamic simulation. In this paper, we introduce a second-order unsplit upwind method of Godunov type for curvilinear coordinates and apply it to gas dynamics simulations involving complex geometries by using overlapping grids.

1 Introduction

Since the appearance of the landmark paper by Harten, Lax and van Leer on upwind finite difference methods and Godunov schemes a decade ago [20], there has been rapid development of higher-order nonlinear upwind finite difference methods for the simulation of compressible hydrodynamics. Their huge success is demonstrated, for example, by the extensive adaptive mesh refinement computations that have been done in recent years using higher-order Godunov methods [1], [7], [8], [31] as well as the treatment given in the recent books of Hirsch [23] and Leveque [25] summarizing much of the recent literature on methods of this type.

In this paper we present an unsplit Godunov method for curvilinear grids and use it with the composite overlapping grid method for representing complex geometries. A composite overlapping grid, also known as a "Chimera" grid, or simply an "overlapping grid", is a set of logically rectangular curvlinear component grids that completely covers a computational region and which overlap where they meet. Overlapping grids have the capability to represent complex geometries within the context of logically rectangular grids. Since each component grid can be relatively smooth, overlapping grids provide an ideal framework in which to use finite difference methods, which tend to perform best on smooth grids. The potential of the overlapping grid method was demonstrated by initial work in the late 1970's and early 1980's of Starius [34], Kreiss [24], and Steger [35]. This potential was borne out by successful 3D aerodynamic simulations involving configurations as complex as the space shuttle [12], and with moving components [19], and also the development by Chesshire and Henshaw of the fully automatic grid overlapping procedure for 2D and 3D grids (CMPGRD) that forms the basis for the present work [9], [14], [16]. Recent work with CMPGRD-generated overlapping grids is reported in [21], [22], [29], [30].

This paper extends the work of Colella [17] and Saltzman [33] on unsplit Godunov methods for multidimensional flows to curvilinear grids. The method introduced here can be viewed as a modification of Colella's quadrilateral grid method [17]. The differences lie mainly in the use of exact differential volumes, rather than exact quadrilateral volumes and in the approximation of the transverse flux terms in the Godunov predictor step.

2 The Difference Scheme

This section presents an "unsplit" Godunov-type method for systems of conservation laws in n_d space dimensions and on curvilinear grids. In n_d-dimensional Cartesian coordinates $x := (x^1, ..., x^{n_d})$, consider the system of n conservation laws given by

$$\partial_t u + \sum_{i=1}^{n_d} \partial_{x^i} f^i(u) = 0. \tag{1}$$

where $u := u(x) : \mathbb{R}^{n_d} \to \mathbb{R}^n$ is the n-vector of unknowns, and $f^i := f^i(u) : \mathbb{R}. \to \mathbb{R}^n$, $i = 1, ..., n$ are the flux vectors. A system of partial differential equations written in this form is said to be in *conservation form*.

The system (1) can be transformed to a curvilinear coordinate system with independent variables $\xi := (\xi^1, \xi^2, \xi^3)$ by using the chain rule. This yields

$$\partial_t u + \sum_{\ell=1}^{n_d} \left(\sum_{i=1}^{n_d} (\partial_{x^i} \xi^\ell) \cdot (\partial_{\xi^\ell} f^i) \right) = 0, \tag{2}$$

which is no longer in conservation form. However it can be put in conservation form by first introducing the functions $\tilde{\xi}^\ell_{x^i} := J_x(\partial_{x^i} \xi^\ell)$, $i = 1, ..., n_d$, where $J_x := det|\frac{\partial x}{\partial \xi}|$ is the determinant of the transformation Jacobian matrix. It can be verified that the identity

$$\sum_{\ell=1}^{n_d} \left(\tilde{\xi}^\ell_{x^i} \cdot \partial_{\xi^\ell} f^i \right) = \sum_{\ell=1}^{n_d} \partial_{\xi^\ell} \left(\tilde{\xi}^\ell_{x^i} \cdot f^i \right) \tag{3}$$

holds by noting that

$$\sum_{\ell=1}^{n_d} \partial_{\xi^\ell} \tilde{\xi}^\ell_{x^i} \equiv 0. \tag{4}$$

Using (3), (2) can then be written in conservation form:

$$J_x \partial_t u + \sum_{\ell=1}^{n_d} \partial_{\xi^\ell} F^\ell(u) = 0 \tag{5}$$

where the transformed fluxes are given by

$$F^\ell(u) := \sum_{i=1}^{n_d} \tilde{\xi}^\ell_{x^i} f^i(u), \quad \tilde{\xi}^\ell_{x^i} := (\partial_{x^i} \xi^\ell)/J_\xi. \tag{6}$$

For computational convenience, (5) can be rewritten as

$$\partial_t u + J_\xi \sum_{\ell=1}^{n_d} \partial_{\xi^\ell} F^\ell(u) = 0 \tag{7}$$

where $J_\xi := det|\frac{\partial \xi}{\partial x}|$. In the form (7), all quantities can be computed from the original flux functions $f^i(u)$, $i = 1, ..., n_d$ using the elements of the transformation Jacobian $\frac{\partial \xi}{\partial x}$.

The system (7) can also be rewritten in *quasilinear form* by starting with (2) and using the relation

$$\partial_{\xi^\ell} f^i(u) = \frac{\partial f^i}{\partial u} \cdot (\partial_{\xi^\ell} u).$$

Then

$$\partial_t u + J_\xi \sum_{\ell=1}^{n_d} A^\ell(u) \partial_{\xi^\ell} u = 0 \tag{8}$$

where

$$A^\ell(u) := \sum_{i=1}^{n_d} \tilde{\xi}^\ell_{x^i} \frac{\partial f^i}{\partial u}. \tag{9}$$

Transforming from one set of dependent variables to another (say, from conservation variables to primitive variables in the case of the Euler equations) is most conveniently accomplished when the system of equations is in quasilinear form. In this paper, conservation variables will be denoted by u and primitive variables by v. If these variables are related by the differential relation

$$du = M dv, \qquad M \in \mathbb{R}^n \times \mathbb{R}^n,$$

then the equations can be written in terms of the primitive variables as

$$\partial_t v + J_\xi \sum_{\ell=1}^{n_d} B^\ell(v) \partial_{\xi^\ell} v = 0 \tag{10}$$

where

$$B^\ell(v) := M^{-1} A^\ell(u) M. \tag{11}$$

This transformation is used in section 3 when computing the eigenstructure of the Jacobian matrices $A^\ell(u)$. The conservative and primitive variables are related directly by a nonlinear transformation of the form

$$u = N(v).$$

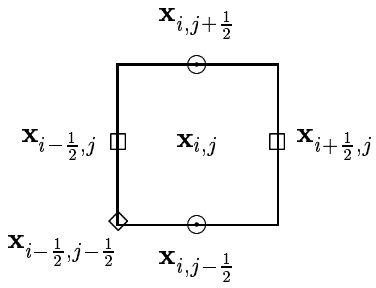


Figure 1: Index definitions

2.1 Notation and Grid Setup

The method that will be introduced in this paper is a cell-centered finite-difference method of Godunov type. It is essentially equivalent to a finite-volume method, although the "volume" and "area" elements, rather than being physical volumes of cells and areas of cell-faces correspond to differential volumes and areas. This is in keeping with the original approach suggested by Chesshire and Henshaw in their paper introducing overlapping grids [14]. It is also possible to derive finite volume methods of the traditional type for use with overlapping grids; this is discussed briefly in section 2.6 below, (see also [17]) although all the computations presented in this paper use the differential form of the volumes and areas, since this information is available as output from the CMPGRD overlapping grid generation program. In transformed coordinates, ξ^ℓ, each component grid of an overlapping grid structure is a tensor product of uniform one-dimensional grids, with some grid cells possibly unused. The notation is illustrated in figure 1 for the two-dimensional case. Cell-centered values are denoted with integer subscripts, while face-centered and vertex-centered values are denoted using both integer and half-integer subscripts. Saltzman [33] introduces a convenient notation that we will employ in this paper. Notation is simplified by using "implied indexing". This means that if an index is missing, it takes some default value. The default values are integer values with the following rule:

$$\mathbf{u} := \mathbf{u}^n_{i,j,k}. \tag{12}$$

An application of this rule is

$$\mathbf{u}^{n+\frac{1}{2}}_{j-1} := \mathbf{u}^{n+\frac{1}{2}}_{i,j-1,k}. \tag{13}$$

An additional convenience is to introduce *index functions* as follows: In three dimensions,

$$\alpha = \alpha(\ell): \quad \alpha(1) = i, \alpha(2) = j, \alpha(3) = k, \tag{14}$$

$$\beta = \beta(\ell): \quad \beta(1) = j, \beta(2) = k, \beta(3) = i, \tag{15}$$

$$\gamma = \gamma(\ell): \quad \gamma(1) = k, \gamma(2) = i, \gamma(3) = j. \tag{16}$$

For two dimensions, α and β are defined in the analogous way.

2.2 The corrector step

The Godunov-type method that will be introduced here can be considered as a predictor-corrector method for advancing an approximate solution of (7) by one timestep [17, 33]. The *second* step (the "corrector" step) of the method (also the conservative step) is given by

$$u^{n+1} = u^n + \sum_{\ell=1}^{n_d} \lambda_\ell \Delta_\ell F_h^\ell({}_{(-)}u_{\alpha+\frac{1}{2}}^{n+\frac{1}{2}}, {}_{(+)}u_{\alpha+\frac{1}{2}}^{n+\frac{1}{2}}) \tag{17}$$

where $\lambda_\ell := J_\xi \Delta t / \Delta \xi^\ell$ and $\Delta_\ell u_{\alpha+\frac{1}{2}} := u_{\alpha+\frac{1}{2}} - u_{\alpha-\frac{1}{2}}$ is the backward difference operator in the ξ^ℓ-direction. The vectors ${}_{(-)}u_{\alpha+\frac{1}{2}}^{n+\frac{1}{2}}$ and ${}_{(+)}u_{\alpha+\frac{1}{2}}^{n+\frac{1}{2}}$ are "left" and "right" states on the cell face centered at $(\alpha + \frac{1}{2}, \beta, \gamma)$. The function $F_h^\ell(u_L, u_R)$ is the approximate flux function. The evaluation of the Jacobian determinant J_ξ can be done in several ways; this is discussed in section 2.6 below. As pointed out by Harten et. al. [20], one can write a Godunov method either in terms of such an approximate flux function or in terms of an approximate (or possibly exact) solution to a one-dimensional Riemann problem for the left and right states u_L and u_R, respectively. Since the method discussed in this paper employs an approximate flux function rather than a Riemann solver, the form (17) is preferred.

2.3 The predictor step

In some sense, the corrector step is the trivial step in the Godunov method. For problems involving more than one space dimension, most of the sophistication, and hence most of the work goes into the computation of the face-centered predicted values ${}_{(\pm)}u_{\alpha+\frac{1}{2}}^{n+\frac{1}{2}}$, $\alpha = i, j, k$, and in the evaluation of the approximate flux function $F_h^\ell(u_L, u_R)$. "Classical" Godunov methods,

such as those discussed in [20], are defined for systems of conservation laws in one space dimension. In extending the concepts to more than one space dimension, there are many options available, particularly in the design of the predictor step. Both Colella [17] and Saltzman [33] introduce predictors that keep as close as possible to the basic upwind spirit of one-dimensional Godunov methods.

While the implementation details of a multidimensional Godunov scheme can seem quite formidable, the basic theme in designing the predictor for an "unsplit" Godunov method is to make sure that the difference scheme computes updated values using previous values from the appropriate (upwind) domain of dependence. Much more detail can be found in Colella [17] and in the book by Leveque [25]. In the method introduced in the present paper, an approach very similar to Collela's [17] is followed.

Given cell-centered values of the solution at timestep n, $u := u^n_{ijk}$, predicted values at face centers can be computed using the first terms in a Taylor's series:

$$ {}_{(\mp)}u^{n+\frac{1}{2}}_{\alpha\pm\frac{1}{2}} = u + \frac{\Delta t}{2}\partial_t u \pm \frac{\Delta\xi^\ell}{2}\partial_{\xi^\ell} u. \tag{18} $$

(All the values on the right-hand side of (18) are evaluated at cell centers at time level n.) The spatial derivative on the right-hand side of (18) can be computed using a difference approximation; the time derivative can be evaluated by first using the differential equation to replace $\partial_t u$ with spatial derivatives of the flux function $F^\ell(u)$ and approximating the resulting expression. Both Colella and Saltzman take the approach of using the differential equation in conservation form (5) and then expanding the flux derivative in the ξ^ℓ-direction so that it is written in terms of $\partial_{\xi^\ell} u$, leaving the transverse derivatives in terms of fluxes. This leads to the formula

$$ \begin{aligned} {}_{(\mp)}u^{n+\frac{1}{2}}_{\alpha\pm\frac{1}{2}} = u \quad & \pm\tfrac{\Delta\xi^\ell}{2}(1 \mp \lambda_\ell A^\ell(u))\partial_{\xi^\ell} u - J_\xi \tfrac{\Delta t}{2}\textstyle\sum_{m\neq l}\partial_{\xi^m} F^m(u) \\ & - J_\xi \tfrac{\Delta t}{2}\textstyle\sum_{i=1}^{n_d}(\partial_{\xi^\ell}\tilde{\xi}^\ell_{x^i})\cdot f^i(u). \end{aligned} \tag{19} $$

Colella approximates $\partial_{\xi^\ell} u$ using a limited central difference. The flux derivatives $\partial_{\xi^m} F^m(u)$ are approximated in the same or similar way as in the corrector step, by replacing them with $\partial_{\xi^m} F^m(u^*)$ and determining u^* using an approximate Riemann solver. This approach retains the spirit of upwinding since the approximate Riemann solve attempts to guarantee that only components of u from the proper domain of dependence are used when differencing the flux functions. This approach is also similar to the one taken

by Saltzman for three-dimensional rectilinear coordinate grids. There is an additional complication in curvilinear coordinates, however, due to the final term on the right-hand side of (19), involving the spatial derivatives of the Jacobian elements $\partial_{\xi^\ell}\tilde{\xi}^\ell_{x^i}$. As Colella points out in [17], care must be taken that these terms be approximated in such a way that pure one-dimensional streaming flows (i.e. flows in which all variables are constant) are preserved. This is accomplished by making sure that ${}_{(\mp)}u^{n+\frac{1}{2}}_{\alpha\pm\frac{1}{2}} \equiv u$ when transverse derivatives ($\partial_{\xi^m} *, m \neq \ell$) are zero. (In the continuous case, this always holds identically.)

In the present method, free-streaming is preserved by using the quasilinear form of the equations (8) instead when replacing $\partial_t u$ in (18). Equation (19) is then replaced with

$$ {}_{(\mp)}u^{n+\frac{1}{2}}_{\alpha\pm\frac{1}{2}} = u \pm \frac{\Delta\xi^\ell}{2}(1 \pm \lambda_\ell A^\ell(u))\partial_{\xi^\ell} u - J_\xi \frac{\Delta t}{2}\sum_{m\neq l} A^m(u)\partial_{\xi^m} u, \qquad (20) $$

which does not contain the additional terms. Since any consistent approximation of ∂_{ξ^ℓ} and ∂_{ξ^m} will map constant functions to zero, free-streaming solutions are trivially preserved by a method of this form. The quantity $\partial_{\xi^\ell} u$ can again be replaced with a limited central difference, but it is desirable to approximate the remaining terms $\sum_{m\neq l} A^m(u)\partial_{\xi^m} u$ in such a way that only values in the proper domain of dependence are used. This is accomplished by using a projection technique.

Colella [17] uses characteristic projection operators when computing the limited "*slopes*" that approximate $\Delta\xi^\ell\partial_{\xi^\ell} u$ in (19) When computing, say, ${}_{(-)}u^{n+\frac{1}{2}}_{\alpha+\frac{1}{2}}$, a locally linearized one-dimensional problem is considered near the cell face centered at $(\alpha+\frac{1}{2}, \beta, \gamma)$. The approximation to $\Delta\xi^\ell\partial_{\xi^\ell} u$ is expanded in terms of the right eigenvectors of the local flux Jacobian $A^\ell(u)$, and then only components of the expansion corresponding to characteristics propagating *towards* the appropriate cell face are kept; the remaining components are set to zero. A similar procedure will be used here to approximate the transverse flux derivative terms.

The eigenvalue problem for the flux Jacobian $A^\ell(u)$ can be written as

$$ A^\ell(u)R^\ell(u) = R^\ell(u)\Lambda^\ell(u) \qquad (21) $$

where $\Lambda^\ell(u) = diag\{\kappa^\ell_1(u), \kappa^\ell_2(u), ..., \kappa^\ell_n(u)\}$ contains the eigenvalues of $A^\ell(u)$ on its diagonal, and the right eigenvectors are the columns of $R^\ell(u)$. The

left eigenvalue problem can be described similarly; the left eigenvectors are the rows of the matrix $L^\ell(u) := R^\ell(u)^{-1}$. Projection operators ${}_{(+)}P^\ell(u)$ and ${}_{(-)}P^\ell(u)$ are defined as follows: Let

$$ {}_{(\pm)}\Pi^\ell(u) := \pm\frac{1}{2}|\Lambda^\ell(u)^{-1}|(\Lambda^\ell(u) \pm |\Lambda^\ell(u)|), $$

where $|\Lambda^\ell(u)| := diag\{|\kappa_1^\ell(u)|, ..., |\kappa_n^\ell(u)|\}$. Then

$$ {}_{(\pm)}P^\ell(u) := R^\ell(u) \cdot {}_{(\pm)}\Pi^\ell(u) \cdot L^\ell(u). \tag{22} $$

Projection operators ${}_{(\pm)}P_v^\ell(v)$ in primitive variable coordinates can be defined analogously by instead using the left and right eigenvectors of $B^\ell(v)$. Then

$$ {}_{(\pm)}P_v^\ell(v) := R_v^\ell(v)_{(\pm)}\Pi^\ell(u)L_v^\ell(v) $$

where the columns of $R_v^\ell(v)$ and the rows of $L_v^\ell(v)$ contain the right and left eigenvectors of $B^\ell(v)$ respectively. A vector w can be expanded in terms of the right eigenvectors of $A^\ell(u)$ as follows:

$$ w = \sum_{k=1}^{n} \alpha_k^\ell r_k^\ell(u) \tag{23} $$

where $\alpha_k^\ell := \ell_k^{\ell^T} \cdot w$ and r_k^ℓ, $\ell_k^{\ell^T}$ are the right and left eigenvectors, respectively, of $A^\ell(u)$. So the projection of u into the space of only those eigenvectors corresponding to positive eigenvalues of $A^\ell(u)$, or equivalently, right-propagating characteristics of the local one-dimensional linearized problem can be written as

$$ {}_{(+)}P^\ell(u)u = \sum_{k \ni \kappa_k^\ell > 0} \alpha_k^\ell r_k^\ell(u). \tag{24} $$

With this notation, an approximation for the transverse flux derivative terms can be written as follows:

$$ \begin{aligned} \Delta\xi^\ell A^\ell(u)\partial_{\xi^\ell} u &\cong A^\ell(u)\{{}_{(-)}P^\ell(u)\Delta_\ell u_{\alpha+1} + {}_{(+)}P^\ell(u)\Delta_\ell u\} \\ &= \{{}_{-}A^\ell(u)\Delta_\ell u_{\alpha+1} + {}_{+}{}_{-}A^\ell(u)\Delta_\ell u\}. \end{aligned} \tag{25} $$

where the projected flux Jacobians ${}_{\pm}A^\ell(u)$ are defined by

$$ {}_{\pm}A^\ell(u) := R^\ell(u)(\Lambda^\ell(u)_{(\pm)}\Pi^\ell(u))L^\ell(u). $$

This can be recognized as a standard first-order upwind formula for these terms. Note, however, that since these terms are multiplied by Δt in (20), they only need be first-order accurate in order for ${}_{(\mp)}u^{n+\frac{1}{2}}_{\alpha\pm\frac{1}{2}}$ to be a second order accurate extrapolation.

The term involving the extrapolation-direction derivatives $\partial_{\xi^\ell} u$ in (20) is approximated in the standard way by first replacing the derivative with a monotonized or "limited" central difference. The standard limiter is the one introduced by vanLeer (see [25]). Its most general formulation for systems of conservation laws is given by Colella in [17] and can be written as

$$\Delta\xi^\ell \partial_{\xi^\ell} u \cong \overline{\Delta_\ell u} := \sum_{k=1}^{n} \hat{\alpha}^\ell_k r^\ell_l(u)$$

where

$$\hat{\alpha}^\ell_k := \begin{cases} min(|\alpha^C_k|, 2|\alpha^L_k|, 2|\alpha^R_k|) sgn(\alpha^C_k), & \text{if } \alpha^L_k \alpha^R_k > 0; \\ 0, & \text{otherwise,} \end{cases}$$

and

$$\begin{aligned} \alpha^C_k &:= \ell^{\ell^T}_k \Delta_{o\ell} u \\ \alpha^{R,L}_k &:= \ell^{\ell^T}_k \cdot (u_{\alpha\pm 1} - u). \end{aligned}$$

A simpler form of the limiter for systems does not involve the decomposition of the approximation to $\Delta\xi^\ell \partial_{\xi^\ell} u$, but instead uses the primitive variables v and limits their differences component by component. This is what is actually used in practice by both Colella [17] and Saltzman [33]. The limiter used is

$$\overline{\Delta_\ell v} := (\overline{\Delta_\ell v^{(1)}}, \overline{\Delta_\ell v^{(2)}}, ..., \overline{\Delta_\ell v^{(n)}})^T \tag{26}$$

where

$$\overline{\Delta_\ell v^{(k)}} := \begin{cases} min(|\Delta_{o\ell} v^{(k)}|, 2|\Delta_\ell v^{(k)}|, 2|\Delta_\ell v^{(k)}_{\alpha+1}|) sgn(\Delta_{o\ell} u^{(k)}), & \text{if } \Delta_\ell u^{(k)} \Delta_\ell u^{(k)}_{\alpha+1} > 0 \ ; \\ 0, & \text{otherwise.} \end{cases}$$

A further modification to the limiter is to discard the components of $\overline{\Delta_\ell v}$ corresponding to characteristics propagating *away* from the cell face. It is convenient to compute ${}_{(\mp)}u^{n+\frac{1}{2}}_{\alpha\pm\frac{1}{2}}$ in two steps, first computing the contribution in the ξ^ℓ -direction, ${}_{(\mp)}\hat{u}^{n+\frac{1}{2}}_{\alpha\pm\frac{1}{2}}$, and then computing the transverse flux

derivative contribution. Performing the entire computation in terms of the primitive variables avoids the evaluation of the flux Jacobian $A^\ell(u)$. Summarizing, and using the fact that ${}_{(\pm)}P_v^\ell(v)$ and $B^\ell(v)$ commute, the predictor step can be written

$$
\begin{aligned}
{}_{(\mp)}\hat{v}_{\alpha\pm\frac{1}{2}}^{n+\frac{1}{2}} &= v \pm \frac{1}{2}{}_{(\pm)}P_v^\ell(v)(1 \mp \lambda_\ell B^\ell(v))\overline{\Delta_\ell v} \qquad (27)\\
{}_{(\mp)}u_{\alpha\pm\frac{1}{2}}^{n+\frac{1}{2}} &= N\left({}_{(\mp)}\hat{v}_{\alpha\pm\frac{1}{2}}^{n+\frac{1}{2}} - \frac{1}{2}\sum_{m\neq l}\lambda_m B^m(v)\{{}_{(-)}P_v^m(v)\Delta_m v_{\beta+1} + {}_{(+)}P_v^m(v)\Delta_m v\}\right)
\end{aligned}
$$

Note that the number of operations required to compute the predicted values ${}_{(\mp)}u_{\alpha\pm\frac{1}{2}}^{n+\frac{1}{2}}$ can be decreased by using a trick that involves subtracting and adding a judiciously chosen reference state u_{ref} in the formula above [17]. Also, (26) can be improved by using a fourth-order formula [33]. Evaluation of the flux Jacobians $B^\ell(v)$ implicitly involves the evaluation of transformation derivatives $\tilde{\xi}_{x^i}^\ell$. The details are discussed below in section 2.6.

2.4 The approximate flux function

The numerical flux function used in the present method is a simplification due to Pao and Saltzman [27] of the approximate flux function proposed by Bell, Colella and Trangenstein [2]. A form of this flux function that is convenient to evaluate computationally is introduced here. Let

$$\sigma := (F^\ell(u_L)^T - F^\ell(u_R)^T)\cdot(u_L - u_R).$$

Then the approximate flux function is given by

$$
F_h^\ell(u_L,u_R) := \begin{cases} F_* - \sum_{k=1}^n\{\beta_k\int_0^1 max(\kappa_k(s),0)ds\}\cdot r_k^\ell(\bar{u}), & \text{if } \sigma<0;\\ F_* + \sum_{k=1}^n\{\beta_k\int_0^1 min(\kappa_k(s),0)ds\}\cdot r_k^\ell(\bar{u}), & \text{otherwise}, \end{cases} \qquad (28)
$$

where $\bar{u} := \frac{1}{2}(u_L + u_R)$,

$$
F_*(u_L,u_R) = \begin{cases} F^\ell(u_R), & \text{if } \sigma<0;\\ F^\ell(u_L), & \text{otherwise,} \end{cases}
$$

and $\kappa_k(s)$ is a function that interpolates linearly between the eigenvalues corresponding to the left and right states:

$$\kappa_k(s) := s\kappa_k(u_R) + (1-s)\kappa_k(u_L),$$

and β_k are the coefficients of the expansion of $u_R - u_L$ in terms of the right eigenvectors of $A^\ell(\bar{u})$, i.e.

$$(\beta_1, .., \beta_n)^T := L^\ell(\bar{u}) \cdot (u_R - u_L).$$

Basically, as with all approximate flux functions, (28) produces a standard upwind formula away from sonic points, but makes a careful transition between left and right states near a sonic point.

Since the integrands that appear in (28) are simple linear functions, an explicit formula can be derived for the approximate flux function. Define

$$\begin{aligned} \kappa_k^1 &:= min(sgn\sigma \cdot \kappa_k(u_L), sgn\sigma \cdot \kappa_k(u_R)) \\ \kappa_k^2 &:= max(sgn\sigma \cdot \kappa_k(u_L), sgn\sigma \cdot \kappa_k(u_R)), \\ s_k^* &:= \begin{cases} \kappa_k^1/(\kappa_k^1 - \kappa_k^2) & \text{if } \kappa_k^1 \neq \kappa_k^2; \\ 1 & \text{otherwise,} \end{cases} \end{aligned} \tag{29}$$

and

$$I_k(s_k^*, u_L, u_R) = \frac{1}{2} max(min(s_k^*, 1), 0) \cdot (min(\kappa_k^1, 0) + min(\kappa_k^2, 0)).$$

Then it can be verified that (28) is equivalent to

$$F_h^\ell(u_L, u_R) = F_*(u_L, u_R) + \sum_{k=1}^{n} \beta_k \cdot I(s_k^*, u_L, u_R) \cdot r_k^\ell(\bar{u}). \tag{30}$$

The representation (30) is simple to evaluate and vectorizes well.

2.5 Alternative Finite-Volume differencing

Another possibility for dealing with volumes and areas in the difference approximation is to replace the differential transformation information with exact or approximately computed cell volumes and areas, as would be used in a traditional finite volume method. The main modification that must be made is to change the corrector formula (17) to

$$u^{n+1} = u^n + \frac{1}{V} \sum_{\ell=1}^{n_d} \Delta_\ell \tilde{F}_h^\ell({}_{(-)}u_{\alpha+\frac{1}{2}}^{n+\frac{1}{2}}, {}_{(+)}u_{\alpha+\frac{1}{2}}^{n+\frac{1}{2}}) \tag{31}$$

where $V = V_{i,j,k}$ is the volume of the cell centered at $\mathbf{x}_{ijk}$, and the approximate flux function $\tilde{F}_h^\ell$ is evaluated using the transformed flux function

$$\tilde{F}^\ell(u) := \mathcal{A}^\ell_{\alpha-\frac{1}{2}} \sum_{i=1}^{n_d} f^i(u) \cdot \hat{n}^\ell_i,$$

where $\mathcal{A}^\ell_{\alpha-\frac{1}{2}}$ is the area of the cell face centered at $\mathbf{x}_{\alpha-\frac{1}{2}}$, and $\hat{\mathbf{n}}^\ell := (\hat{n}^\ell_1, \hat{n}^\ell_2, \hat{n}^\ell_3)$ is the unit normal to that cell face. Corresponding modifications must be made to the evaluation of the flux Jacobians $B^\ell(v)$ and transformation matrices R^ℓ, L^ℓ when they are evaluated during the computation of the approximate flux function. The cell areas and volumes can be computed explicitly from the (usually piecewise linear) representation of the grid coordinate system.

2.6 Overlapping grids

An overlapping grid consists of a set of n_g curvilinear logically rectangular "component" grids that cover the computational domain and overlap where they meet. Examples of some overlapping grids are shown in section 5 below. Each computational grid is a tensor-product grid in the transformed coordinates ξ with uniform grid spacings $(\Delta\xi^1_{k_g}, ..., \Delta\xi^{n_d}_{k_g})$, $k_g = 1, ..., n_g$. On each grid, each cell is indexed by coordinates and grid number $(\mathbf{i}, k_g) := (i^1, ..., i^{n_d}, k_g)$, with the interior discretization cells numbered in the range $M_{k_g,\ell,1} \le i^\ell \le M_{k_g,\ell,2}$, $k_g = 1, ..., n_g$, $\ell = 1, ..., n_d$. In addition there are two rows of auxiliary "fictitious" or "ghost" cells added around the boundary of each grid that are used in the boundary condition computation for each grid. These are needed because the effective size of the finite difference stencil for the second-order Godunov method described in this paper is 5×5 and so the central point requires values at the previous timestep up to a distance 2 cells away. In regions where the component grids overlap, some of the cells are used for interpolation of solution values from other component grids, and some may be unneeded for the finite difference computation. There is an integer array $kr_{\mathbf{i},k_g}$ that is used to classify the cells with the rules

$$kr_{\mathbf{i},k_g} = \begin{cases} -k_{int} & \text{if the value in cell } (\mathbf{i}, k_g) \text{ is interpolated from grid } k_{int}; \\ 0 & \text{if cell } (\mathbf{i}, k_g) \text{ is unused ;} \\ k_g & \text{if cell } (\mathbf{i}, k_g) \text{ is a discretization cell} \end{cases}$$

The overlapping grids used in this paper are constructed using the CMPGRD code of Chesshire and Henshaw [11],[14]. To benefit from the flexibility of overlapping grids, it is important to write a PDE solver that can handle

overlapping grids with any number of component grids and with arbitrary configurations of boundary types. The DSK scientific database management system of Chesshire [13] provides a convenient framework in which to accomplish this. It provides data storage allocation for the complex overlapping grid data structures within the Fortran language.

When CMPGRD is used to generate an overlapping grid, it provides all of the information needed to use that grid for a finite difference computation, including the grid coordinates $\mathbf{x}_{\mathbf{i},k_g}$, the transformation Jacobian elements $\partial_{x^i}\xi^\ell$ evaluated at cell vertices $(i-\frac{1}{2}, j-\frac{1}{2}, k-\frac{1}{2})$, the "kr" array $kr_{\mathbf{i},k_g}$ which classifies each cell on the overlapping grid, and lists of interpolation points on each grid and the grids they interpolate from. (More details can be found in [10],[11]). The transformation Jacobian elements are used in several places within the Godunov method described in previous sections; since they are often required either at cell centers or along cell edges, they must be computed from the CMPGRD data which is given only at the cell vertices. Since the finite difference mesh is uniform in the ξ coordinate system, this is easily done by using standard central second-order averaging formulas.

2.7 Updating the Solution on an Overlapping Grid

Since this finite difference method is explicit, the process of updating the PDE solution at each timestep on the overlapping grids is quite similar to the algorithm for a single grid. The Godunov scheme is applied to the interior cells of each grid separately. Boundary conditions on grid sides corresponding to physical boundaries are then updated. When all regular cells on all grids have been updated, the interpolation cells are updated between component grids following the rules that CMPGRD provides. In the present code, biquadratic interpolation is used to interpolate the state variables from a stencil on one component grid to each interpolation point on each component grid. This form of intergrid interpolation is not discretely conservative in the usual sense, since fluxes between component grids are not balanced explicitly. However, experience with computations indicates that this lack of conservation usually does not degrade the solutions; indeed, none of the large scale computations reported in [3],[12],[19] use a conservative interpolation approach. This is perhaps not surprising if one considers the following argument. For smooth solutions, the standard interpolation procedure is approximately conservative since it follows that if the solution is approximated to some order, then integrals of that solution must also

be approximated to some order. It is only when a near-discontinuity such as a shock passes through an interpolation interface that the conservative property is lost. It seems reasonable that for a shock moving at finite speed, the effect of the interaction with the interface will be felt only locally and for a few timesteps, and so only a small perturbation of the solution will result. This is consistent with the typical observation in our calculations that moving shocks apparently do not interact with interpolation interfaces in any significant way. The subject of conservative intergrid interpolation is a complex one, and beyond the scope of this paper. Recent papers in the literature discuss this issue in depth, providing some options for handling this problem [4], [6], [15], [28]. Special cases where conservative interpolation is apparently important have been observed, particularly the case when very slow shocks interact with a grid overlap region in which the mesh cell size changes by a large ($\approx 2-4$) factor [5], [28]. The discussion of this situation in [28] is quite interesting.

3 The Euler Equations

Most upwind schemes require detailed knowledge of the eigenstructure of the flux Jacobians $A^\ell(\mathbf{u})$, $\ell = 1, 2, 3$. The signs of the eigenvalues determine the local one-dimensional characteristic directions, and the left and right eigenvectors are needed to compute the appropriate projections. In this section, the eigenstructure of $A^\ell(\mathbf{u})$ is presented in detail for the Euler equations of gas dynamics in curvilinear coordinates.

It will simplify the notation in this section to let $\mathbf{x} = (x^1, x^2, x^3) = (x, y, z)$ and $\xi = (\xi^1, \xi^2, \xi^3) = (\xi, \eta, \zeta)$. Both representations for $\mathbf{x}$ and ξ will be used interchangeably. Also, in this section, vectors will be denoted with bold face to distinguish them from scalars. Thus the dependent variables are denoted by $\mathbf{u}$ and $\mathbf{v}$; and $\mathbf{x}$ refers to the vector Cartesian coordinate, while x refers to the coordinate x^1. Also, the Cartesian velocity components will be denoted either by (u, v, w), or equivalently, (u^1, u^2, u^3).

Recall that the conservative flux Jacobians $A^\ell(\mathbf{u})$ are related to the primitive variable equation coefficient matrices $B^\ell(\mathbf{v})$ by the similarity transformation of equation (11). Denote by $R_v^\ell(\mathbf{v})$ the matrix whose columns are the right eigenvectors of $B^\ell(\mathbf{v})$, i.e.

$$B^\ell(\mathbf{v}) R_v^\ell(\mathbf{v}) = R_v^\ell(\mathbf{v}) \Lambda^\ell(\mathbf{u}).$$

The eigenvalues κ_k^ℓ, $k = 1, ..., n$ of $A^\ell(\mathbf{u})$ and $B^\ell(\mathbf{v})$ are the same and the

transformation matrices are related by

$$MR_v^\ell(\mathbf{v}) = R^\ell(\mathbf{u}).$$

It is simpler to compute R_v^ℓ and then compute R^ℓ using the similarity transformation matrix M than to compute R^ℓ directly.

Following Hirsch [23], chapter 16, the conservative form of the Euler equations in three space dimensions is written as

$$\partial_t \mathbf{u} + \sum_{i=1}^{n_d} \partial_{x^i} \mathbf{f}^i = 0 \tag{32}$$

where n_d is the number of space dimensions, and for $n_d = 3$,

$$\mathbf{u} = (\rho, \rho u, \rho v, \rho w, \rho E)^T \tag{33}$$

$$\mathbf{f}^i = (\rho u^i, \rho u^i u^1 + \delta_{i1} p, \rho u^i u^2 + \delta_{i2} p, \rho u^i u^3 + \delta_{i3} p, \rho u^i H)^T \tag{34}$$

with $H := E + p/\rho$. An equation of state is required to complete this system. To obtain the Euler equations in two dimension ($n_d = 2$), the fourth variable and equation are removed; for $n_d = 1$, the third variable and equation are removed also.

In curvilinear coordinates, the equations become

$$\partial_t \mathbf{u} + J_\xi \sum_{\ell=1}^{n_d} \partial_{\xi^\ell} \mathbf{F}^\ell(\mathbf{u}) = 0 \tag{35}$$

where

$$\mathbf{F}^\ell(\mathbf{u}) =: \begin{pmatrix} \rho U^\ell \\ u\rho U^\ell + \tilde{\xi}_x^\ell p \\ v\rho U^\ell + \tilde{\xi}_y^\ell p \\ w\rho U^\ell + \tilde{\xi}_z^\ell p \\ U^\ell(\rho E + p) \end{pmatrix}, \ell = 1, 2, 3 \tag{36}$$

where we have introduced the contravariant velocities

$$U^\ell := \sum_{i=1}^{n_d} \tilde{\xi}_{x^i}^\ell u^i.$$

In terms of the primitive variables and in curvilinear coordinates, the Euler equations can be written as

$$\partial_t \mathbf{v} + J_\xi \sum_{\ell=1}^{n_d} B^\ell(\mathbf{v}) \partial_{\xi^\ell} \mathbf{v} = 0. \tag{37}$$

Here,

$$B_\ell(\mathbf{v}) := \begin{pmatrix} U^\ell & \tilde{\xi}^\ell_x\rho & \tilde{\xi}^\ell_y\rho & \tilde{\xi}^\ell_z\rho & 0 \\ 0 & U^\ell & 0 & 0 & \tilde{\xi}^\ell_x/\rho \\ 0 & 0 & U^\ell & 0 & \tilde{\xi}^\ell_y/\rho \\ 0 & 0 & 0 & U^\ell & \tilde{\xi}^\ell_z/\rho \\ 0 & \tilde{\xi}^\ell_x\rho c^2 & \tilde{\xi}^\ell_y\rho c^2 & \tilde{\xi}^\ell_z\rho c^2 & U^\ell \end{pmatrix}, \ell = 1,2,3$$

where c is the sound speed defined by $c^2 := \frac{\partial p}{\partial \rho}|_S$. For the purposes of this paper, a perfect gas is assumed; then $c^2 = \gamma p/\rho$. The primitive variables $\mathbf{v}$ are given by $\mathbf{v} := (\rho, u, v, w, p)^T$, where $p = (\gamma - 1)(\rho E - \rho q^2/2)$, and $q^2 := \sum_{i=1}^{n_d} (u^i)^2$. For a perfect gas, H can be reexpressed in terms of c and q:

$$H = \frac{c^2}{\gamma - 1} + \frac{q^2}{2}. \tag{38}$$

The transformation matrix relating $A^\ell(\mathbf{u})$ and $B^\ell(\mathbf{v})$ is given by

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ u & \rho & 0 & 0 & 0 \\ v & 0 & \rho & 0 & 0 \\ w & 0 & 0 & \rho & 0 \\ \frac{q^2}{2} & \rho u & \rho v & \rho w & \frac{1}{\gamma - 1} \end{pmatrix}. \tag{39}$$

It is convenient to define a scaled version of the transformation elements and of the contravariant velocity by

$$\begin{aligned} \hat{\xi}^\ell_{x^i} &:= \tilde{\xi}^\ell_{x^i}/|\tilde{\xi}^\ell_{\mathbf{x}}|, \\ \hat{U}^\ell &:= U^\ell/|\tilde{\xi}^\ell_{\mathbf{x}}| = \sum_{i=1}^{n_d} \hat{\xi}^\ell_{x^i} u^i \end{aligned} \tag{40}$$

where

$$|\tilde{\xi}^\ell_{\mathbf{x}}| := \left(\sum_{i=1}^{n_d} (\tilde{\xi}^\ell_{x^i})^2 \right)^{\frac{1}{2}}.$$

With this notation, the eigenvalues of $B^\ell(\mathbf{v})$ (and of $A^\ell(\mathbf{u})$) are given by

$$|\tilde{\xi}^\ell_{\mathbf{x}}|\{\hat{U}^\ell - c, \hat{U}^\ell, \hat{U}^\ell, \hat{U}^\ell, \hat{U}^\ell + c\} \tag{41}$$

A conveniently scaled form of the matrix of right eigenvectors of $B^\ell(\mathbf{v})$ is given by

$$R_v^\ell(\mathbf{v}) = \begin{pmatrix} \rho/2c & \hat{\xi}_x^\ell & \hat{\xi}_y^\ell & \hat{\xi}_z^\ell & \rho/2c \\ -\hat{\xi}_x^\ell/2 & 0 & -\hat{\xi}_z^\ell & \hat{\xi}_y^\ell & \hat{\xi}_x^\ell/2 \\ -\hat{\xi}_y^\ell/2 & \hat{\xi}_z^\ell & 0 & -\hat{\xi}_x^\ell & \hat{\xi}_y^\ell/2 \\ -\hat{\xi}_z^\ell/2 & -\hat{\xi}_y^\ell & \hat{\xi}_x^\ell & 0 & \hat{\xi}_z^\ell/2 \\ c\rho/2 & 0 & 0 & 0 & c\rho/2 \end{pmatrix} \tag{42}$$

where here, the jth column is the right eigenvector corresponding to the jth eigenvalue in the list (41). The inverse matrix, containing as its rows the left eigenvectors of $B^\ell(\mathbf{v})$ is given by

$$R_v^\ell(\mathbf{v})^{-1} = \begin{pmatrix} 0 & -\hat{\xi}_x^\ell & -\hat{\xi}_y^\ell & -\hat{\xi}_z^\ell & 1/c\rho \\ \hat{\xi}_x^\ell & 0 & \hat{\xi}_z^\ell & -\hat{\xi}_y^\ell & -\hat{\xi}_x^\ell/c^2 \\ \hat{\xi}_y^\ell & -\hat{\xi}_z^\ell & 0 & \hat{\xi}_x^\ell & -\hat{\xi}_y^\ell/c^2 \\ \hat{\xi}_z^\ell & \hat{\xi}_y^\ell & -\hat{\xi}_x^\ell & 0 & -\hat{\xi}_z^\ell/c^2 \\ 0 & \hat{\xi}_x^\ell & \hat{\xi}_y^\ell & \hat{\xi}_z^\ell & 1/c\rho \end{pmatrix}. \tag{43}$$

Computing $R^\ell(\mathbf{u})$ and $R^\ell(\mathbf{u})^{-1}$ is in principle just a simple matter of multiplying by M and M^{-1}, respectively. The resulting expressions are given below. For the three space dimensional case, the transformation matrices are

$$R^\ell(\mathbf{u}) =$$

$$\begin{pmatrix} 1 & \hat{\xi}_x^\ell & \hat{\xi}_y^\ell & \hat{\xi}_z^\ell & 1 \\ u - c\hat{\xi}_x^\ell & \hat{\xi}_x^\ell u & \hat{\xi}_y^\ell u - \rho\hat{\xi}_z^\ell & \hat{\xi}_z^\ell u + \rho\hat{\xi}_y^\ell & u + c\hat{\xi}_x^\ell \\ v - c\hat{\xi}_y^\ell & \hat{\xi}_x^\ell v + \rho\hat{\xi}_z^\ell & \hat{\xi}_y^\ell v & \hat{\xi}_z^\ell v - \rho\hat{\xi}_x^\ell & v + c\hat{\xi}_y^\ell \\ w - c\hat{\xi}_z^\ell & \hat{\xi}_x^\ell w - \rho\hat{\xi}_y^\ell & \hat{\xi}_y^\ell w + \rho\hat{\xi}_x^\ell & \hat{\xi}_z^\ell w & w + c\hat{\xi}_z^\ell \\ H - c\hat{U}^\ell & \rho(\hat{\xi}_z^\ell v - \hat{\xi}_y^\ell w) + \hat{\xi}_x^\ell \frac{q^2}{2} & \rho(\hat{\xi}_x^\ell w - \hat{\xi}_z^\ell u) + \hat{\xi}_y^\ell \frac{q^2}{2} & \rho(\hat{\xi}_y^\ell u - \hat{\xi}_x^\ell v) + \hat{\xi}_z^\ell \frac{q^2}{2} & H + c\hat{U}^\ell \end{pmatrix}, \tag{44}$$

and

$$R^\ell(\mathbf{u})^{-1} =$$

$$\frac{\gamma-1}{2c^2}\begin{pmatrix} \frac{q^2}{2}+\frac{c\hat{U}^\ell}{\gamma-1} & -\frac{c\hat{\xi}_x^\ell}{\gamma-1}-u & -\frac{c\hat{\xi}_y^\ell}{\gamma-1}-v & -\frac{c\hat{\xi}_z^\ell}{\gamma-1}-w & 1 \\ \frac{2c^2(\hat{\xi}_y^\ell w-\hat{\xi}_z^\ell v)}{(\gamma-1)\rho}-2\hat{\xi}_x^\ell(q^2+\frac{c^2}{\gamma-1}) & 2\hat{\xi}_x^\ell u & 2\hat{\xi}_x^\ell v+\frac{2c^2\hat{\xi}_z^\ell}{(\gamma-1)\rho} & 2\hat{\xi}_x^\ell w-\frac{2c^2\hat{\xi}_y^\ell}{(\gamma-1)\rho} & -2\hat{\xi}_x^\ell \\ \frac{2c^2(\hat{\xi}_z^\ell u-\hat{\xi}_x^\ell w)}{(\gamma-1)\rho}-2\hat{\xi}_y^\ell(q^2+\frac{c^2}{\gamma-1}) & 2\hat{\xi}_y^\ell u-\frac{2c^2\hat{\xi}_z^\ell}{(\gamma-1)\rho} & 2\hat{\xi}_y^\ell v & 2\hat{\xi}_y^\ell w+\frac{2c^2\hat{\xi}_x^\ell}{(\gamma-1)\rho} & -2\hat{\xi}_y^\ell \\ \frac{2c^2(\hat{\xi}_x^\ell v-\hat{\xi}_y^\ell u)}{(\gamma-1)\rho}-2\hat{\xi}_z^\ell(q^2+\frac{c^2}{\gamma-1}) & 2\hat{\xi}_z^\ell u+\frac{2c^2\hat{\xi}_y^\ell}{(\gamma-1)\rho} & 2\hat{\xi}_z^\ell v-\frac{2c^2\hat{\xi}_x^\ell}{(\gamma-1)\rho} & 2\hat{\xi}_z^\ell w & -2\hat{\xi}_z^\ell \\ \frac{q^2}{2}-\frac{c\hat{U}^\ell}{\gamma-1} & \frac{c\hat{\xi}_x^\ell}{\gamma-1}-u & \frac{c\hat{\xi}_y^\ell}{\gamma-1}-v & \frac{c\hat{\xi}_z^\ell}{\gamma-1}-w & 1 \end{pmatrix}. \tag{45}$$

In two space dimensions, a convenient form for the matrix of right eigenvectors is

$$R_4^\ell(\mathbf{u}) = \begin{pmatrix} 1 & 1/c & 0 & 1 \\ u-\hat{\xi}_x^\ell c & u/c & -\hat{\xi}_y^\ell & u+\hat{\xi}_x^\ell c \\ v-\hat{\xi}_y^\ell c & v/c & \hat{\xi}_x^\ell & v+\hat{\xi}_y^\ell c \\ \frac{q^2}{2}+\frac{c^2}{\gamma-1}-c\hat{U}^\ell & q^2/2c & \hat{\xi}_x^\ell v-\hat{\xi}_y^\ell u & \frac{q^2}{2}+\frac{c^2}{\gamma-1}+c\hat{U}^\ell \end{pmatrix} \tag{46}$$

and the left eigenvectors are the rows of

$$R_4^\ell(\mathbf{u})^{-1} =$$

$$\begin{pmatrix} \frac{\hat{U}^\ell}{2c}+\frac{(\gamma-1)q^2}{4c^2} & -\left(\frac{\hat{\xi}_x^\ell}{2c}+\frac{(\gamma-1)u}{2c^2}\right) & -\left(\frac{\hat{\xi}_y^\ell}{2c}+\frac{(\gamma-1)v}{2c^2}\right) & \frac{\gamma-1}{2c^2} \\ c-(\gamma-1)\frac{q^2}{2c} & (\gamma-1)\frac{u}{c} & (\gamma-1)\frac{v}{c} & -\frac{\gamma-1}{c} \\ -\hat{\xi}_x^\ell v+\hat{\xi}_y^\ell u & -\hat{\xi}_y^\ell & \hat{\xi}_x^\ell & 0 \\ -\frac{\hat{U}^\ell}{2c}+\frac{(\gamma-1)q^2}{4c^2} & \left(\frac{\hat{\xi}_x^\ell}{2c}-\frac{(\gamma-1)u}{2c^2}\right) & \left(\frac{\hat{\xi}_y^\ell}{2c}-\frac{(\gamma-1)v}{2c^2}\right) & \frac{\gamma-1}{2c^2} \end{pmatrix}. \tag{47}$$

4 Boundary Conditions for Euler Flow

In the overlapping grid method, the computational region is covered with a set of component grids that overlap where they meet and which accurately represent all physical boundaries in the problem. The edges of each logically rectangular or hexahedral component grid are either used to represent part of the physical boundary of the computational region, an artificial boundary

in the problem, or are part of a region of overlap in which boundary cells are updated with an interpolation formula as described in section 2.6. In this section, the implementation of the boundary conditions on the edges corresponding to non-overlap boundaries is discussed. The implementation details will be given for the two-dimensional case only; it is straight-forward to generalize the boundary conditions to the three dimensional case.

The non-overlap boundary conditions in the code are implemented by using rows of "fictitious" or "ghost" cells outside the edge of the interior computational domain. The values in the ghost cells are set so that when cells near the boundary are updated with and interior difference stencil involving some of the ghost cells, the boundary conditions are approximated to some order of accuracy. In a Godunov code, it is convenient to choose the ghost cell values so that reasonable end states will be provided to the Riemann solver or approximate flux evaluation along the boundary edges of the grid. We will discuss this procedure for three types of non-overlap boundaries: slip surfaces, outflow boundaries, and general far-field boundaries (including inflow boundaries).

In order to provide values for the approximate flux function evaluation on boundary edges, the second-order Godunov method requires two rows of ghost cell values outside the boundaries, plus each of the four cells diagonally outside the corners of the grid. For the case when no cells along the boundary have been removed in the grid construction process, this set of ghost cells is defined by

$$M_{k_g,\ell,1} \leq \beta \leq M_{k_g,\ell,2}, k_g = 1, ..., n_g, \ell = 1, 2, (\beta = j, i)$$

for

$$\alpha = M_{k_g,\ell,1} - 2, M_{k_g,\ell,1} - 1, M_{k_g,\ell,2} + 1, M_{k_g,\ell,2} + 2, \alpha = i, j$$

and

$$\begin{aligned}(i,j) \quad = \quad & (M_{k_g,\ell,1} - 1, M_{k_g,\ell,2} - 1), (M_{k_g,\ell,1} - 1, M_{k_g,\ell,2} + 1), \\ & (M_{k_g,\ell,1} + 1, M_{k_g,\ell,2} - 1), (M_{k_g,\ell,1} + 1, M_{k_g,\ell,2} + 1)\end{aligned}$$

In the case when some cells near the boundary of the region have been removed, the required ghost cells are determined by inspecting each interior cell near the boundary and determining which values are needed by the discretization stencil. Each of the primitive variables ρ, u, v and p must be assigned a value in order to provide endstates u_L and u_R for the approximate flux function evaluation.

4.1 Solid Walls

On sides corresponding to solid walls, the only condition that needs to be specified for well-posedness of the Euler equations is the slip surface condition $\mathbf{u} \cdot \hat{\mathbf{n}} = 0$. This condition can set only one of the four variables needed for the approximate flux function evaluation. The remaining values must be assigned consistently with this condition so as to maintain well-posedness. A condition on the normal derivative of the pressure follows directly from the slip wall condition. Most Euler codes simply set $\hat{\mathbf{n}} \cdot \nabla p = 0$. However, as pointed out by Rizzi [32], on a general curvilinear grid, this is only a first-order accurate condition. A more accurate condition is given by

$$\hat{\mathbf{n}} \cdot \nabla p = \mathbf{u} \cdot (\mathbf{u} \cdot \nabla)\hat{\mathbf{n}} = 0, \tag{48}$$

which corrects the normal pressure derivative to account for the acceleration of the fluid due to the curved surface of the solid wall. Rizzi simplifies this condition by applying the normal velocity boundary condition and writes the pressure condition in curvilinear coordinates, giving

$$\alpha \frac{\partial p}{\partial \eta} = -\beta \frac{\partial p}{\partial \xi} + U^{\xi} \left(u \frac{\partial \eta_x}{\partial \xi} + v \frac{\partial \eta_y}{\partial \xi} \right) \tag{49}$$

where ξ is the curvilinear coordinate along the boundary, η is the (not necessarily orthogonal) coordinate moving away from the boundary, subscripted variables are transformation Jacobian elements, $U^{\xi} := \xi_x u + \xi_y v$, $\alpha = {\eta_x}^2 + {\eta_y}^2$, and $\beta = \eta_x \xi_x + \eta_y \xi_y$. A differenced form of this boundary condition is used to obtain pressure values in the ghost cells at the boundary of the grid.

Following arguments in Courant and Friedrichs [18],p. 297, the additional boundary values can be determined by considering the equivalent local reflection problem at the boundary. It follows from this argument that density and tangential velocity must be continuous across the boundary, and so these values are set by imposing discrete continuity conditions on ρ and $\hat{\mathbf{t}} \cdot \mathbf{u}$. Once the four values are set for each ghost cell, the approximate flux function can be evaluated along the domain boundaries.

Consider the left-hand boundary of component grid k_g and for simplicitly let $M_{k_g,1,1} = 1$, $M_{k_g,2,1} = 1$, $M_{k_g,2,2} = M$. The first and second rows of ghost cells are indexed $(0, j)$, $(-1, j)$, $j = 1, ..., M$, respectively; the first row of cells inside the boundary are indexed $(1, j)$, $j = 1, ..., M$. The boundary itself is at $i = \frac{1}{2}$. Denote by $\hat{\mathbf{n}}$, a normal to this side, and by $\hat{\mathbf{t}}$ a tangential vector. To impose $\hat{\mathbf{n}} \cdot \mathbf{u} = 0$, we must first compute $\hat{\mathbf{n}} \cdot \mathbf{u}$ in terms of the

available grid transformation information. We define normal and tangential vectors by

$$\begin{aligned} \hat{\mathbf{t}} &= (\eta_y, -\eta_x) \\ \hat{\mathbf{n}} &= (\eta_x, \eta_y). \end{aligned}$$

We define normal and tangential velocities U^N and U^T by

$$\begin{aligned} U^N_{ij} &= \eta_{x,ij} u_{ij} + \eta_{y,ij} v_{ij} \\ U^T_{ij} &= \eta_{y,ij} u_{ij} - \eta_{x,ij} v_{ij} \end{aligned}$$

at gridpoint (i, j). (The scaling of these velocities is unimportant since the boundary conditions are homogeneous). Then the boundary conditions for ρ, u and v are given by

$$\begin{aligned} \rho_{0,j} &= \rho_{1,j}; && (50) \\ \rho_{-1,j} &= \rho_{2,j} \\ U^N_{0,j} &= -U^N_{1,j}; && (51) \\ U^N_{-1,j} &= -U^N_{2,j} \\ U^T_{0,j} &= U^T_{1,j}; && (52) \\ U^T_{-1,j} &= U^T_{2,j}. && (53) \end{aligned}$$

The pressure condition (49) is approximated with divided differences. The condition for $p_{0,j}$ is

$$\begin{aligned} \alpha_{\frac{1}{2},j} \frac{\Delta\xi}{\Delta\eta} (p_{1,j} - p_{0,j}) &+ \frac{1}{2} \beta_{\frac{1}{2},j} (p_{1,j+1} - p_{1,j-1}) && (54) \\ &= U^{\xi}_{\frac{1}{2},j} \left(u_{\frac{1}{2},j} (\eta_{x,\frac{1}{2},j+\frac{1}{2}} - \eta_{x,\frac{1}{2},j-\frac{1}{2}}) + v_{\frac{1}{2},j} (\eta_{y,\frac{1}{2},j+\frac{1}{2}} - \eta_{y,\frac{1}{2},j-\frac{1}{2}}) \right), \end{aligned}$$

where $\alpha_{i,j}$ and $\beta_{i,j}$ are the coefficients in (49) above. Values in this formula not available at the appropriate points on the grid are determined using central averages of available values. The approximation of the $dp/d\xi$ in this formula is only first order since it is not centered on the boundary line. It is often the case, however, that the grid is orthogonal or nearly orthogonal at the boundary, in which case this term vanishes or is very small. A similar formulais used to determine $p_{-1,j}$.

4.2 Far-field and Outflow Boundary Conditions

The approach we use to implement boundary conditions for artificial boundaries is to consider a locally one-dimensional problem normal, or approximately normal to the boundary, and decompose the solution into incoming and outgoing characteristic variables using the projection operators discussed in section 2. Incoming variables are set to "free-stream" values, specified as part of the problem, while outgoing varables are simply extrapolated by setting the first difference in the normal direction to zero. Using overlapping grids allows the flexibility to choose grids that are orthogonal near far-field boundaries, which simplifies the general specification of these boundary conditions. This boundary condition is a relatively standard one for compressible flow solvers, see e.g. [26]. This is a reasonably robust approach except in the case when a shock impinges on the boundary. The difficulty here is that the number of incoming characteristics at the boundary changes as the shock impinges, making it difficult to specify the boundary conditions at that time. To avoid this case, it is convenient to specify simple outflow boundary conditions in which the first difference of all state variables is set to zero at the outflow boundary.

4.3 Corner Boundary Conditions

Specifying boundary conditions at corners of grids can be problematic, especially since actual physical corners in a geometry can result in singularities in the flow. The proper resolution of physical corners is beyond the scope of this paper; instead we will consider the simpler cases of corners between physical and artificial boundaries, and discuss an ad-hoc solution to the physical corner boundary problem.

When there is a corner between a physical (i.e. no-slip) boundary and an artificial (inflow, ouflow, far-field) boundary, the presumption is made that the physical situation being modeled is of a solid boundary that continues outside the computational region in some uniform fashion. The appropriate boundary condition to apply for the corresponding corner cell is therefore the no-slip boundary condition. If the artificial boundary ghost cells are updated first, then the no-slip boundary conditions can be applied it the corner cell in the same way as for any regular boundary cell.

Specifying boundary conditions for a corner cell at the intersection of two no-slip boundaries is less clear. The common convention is to assign a normal vector that is the average of the normals on the adjacent sides, and

then approximate the no-slip boundary conditions in a consistent fashion. Since the curvature of the boundary at this point is undefined, it is simplest to $\hat{\mathbf{n}} \cdot \nabla p = 0$, rather than Rizzi's more complicated boundary conditions.

At a corner between two artificial boundaries, again a normal vector can be assigned, and boundary conditions can be implemented with respect to that vector. With overlapping grids, however, it is usually simpler to arrange that artificial boundaries do not intersect at corners. For example in a problem involving exterior flow around some object, the grids defining the object can be embedded in a large circular region rather than a large rectangular region, thus avoiding the question of specifying artificial boundary conditions at a corner.

5 Computational examples

As a numerical example demonstrating the method we present a computation showing the interaction of a shock travelling down a channel with bumps on the wall of the channel. Figure 2 shows the overlapping grid used for the computations. It consists of three component grids: two semi-annular grids with 98x16 cells, and an underlying rectangular grid with 105x70 cells. The remaining figures show the solution at various times. The initial conditions are a step function with discontinuity located at $x = -.85$, to the left of the bumps. The discontinuity satisfies a Riemann problem in which the solution is a right-propagating shock of speed approximately 1.61245 and no other waves. The values of the constant states on either side of the jump are given by

$$\begin{aligned} \rho_L &= 1.625, \quad u_L = 1.00778, \quad v_L = 0.0, \quad p_L = 2.0, \\ \rho_R &= 1.000, \quad u_R = 0.0, \quad v_R = 0.0, \quad p_R = 1.0 \quad . \end{aligned} \tag{55}$$

The irregularities in the contour lines for the initial conditions are a result of the contour plotting package used; the lower portion of the initial shock profile crosses into the lower semi-annular grid. The boundary conditions used are of far-field type at the left boundary, outflow at the right boundary, and general non-slip conditions on all other surfaces.

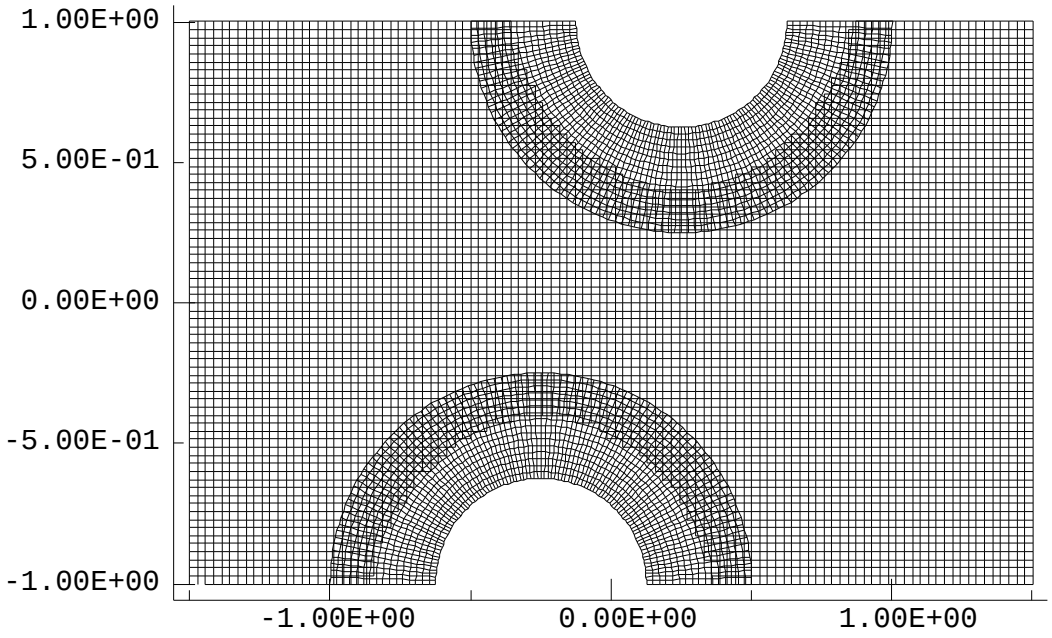


Figure 2: The overlapping grid used for the computations.

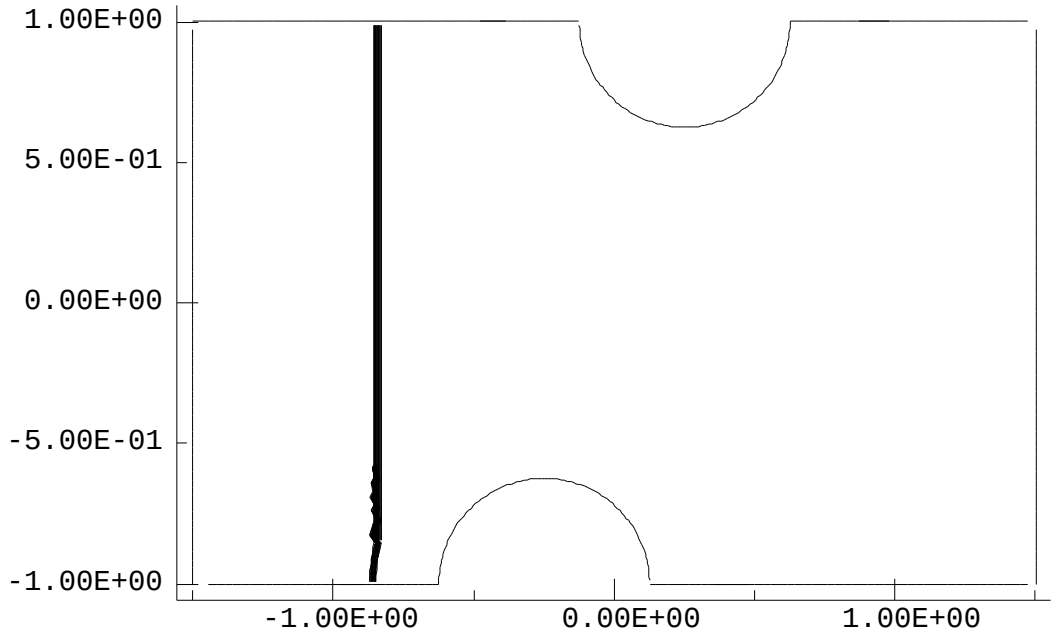


Figure 3: Fluid density at time 0.

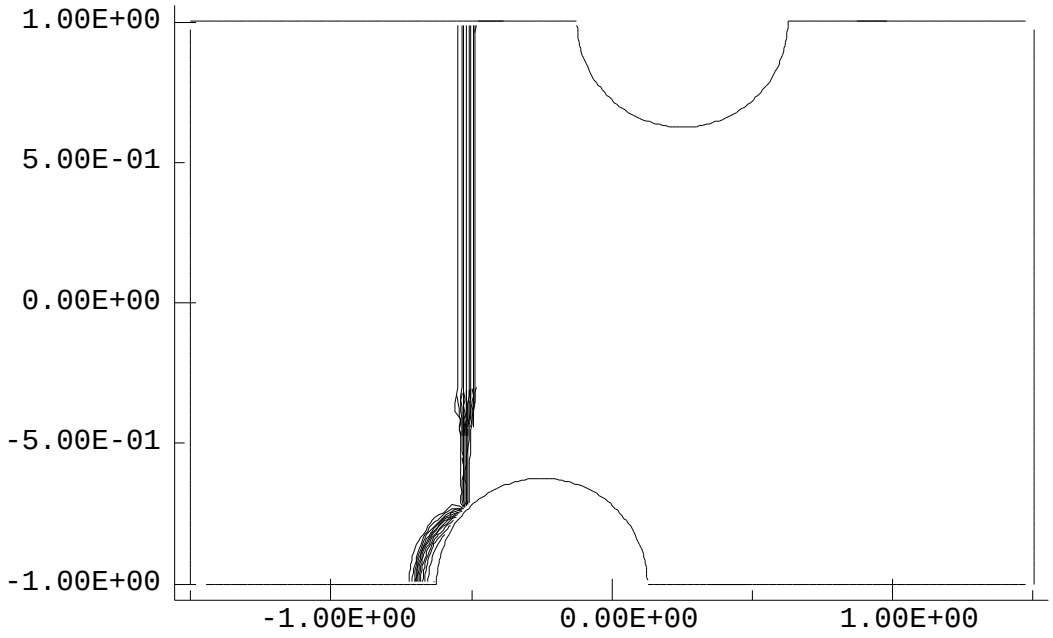


Figure 4: Fluid density at time 0.2.

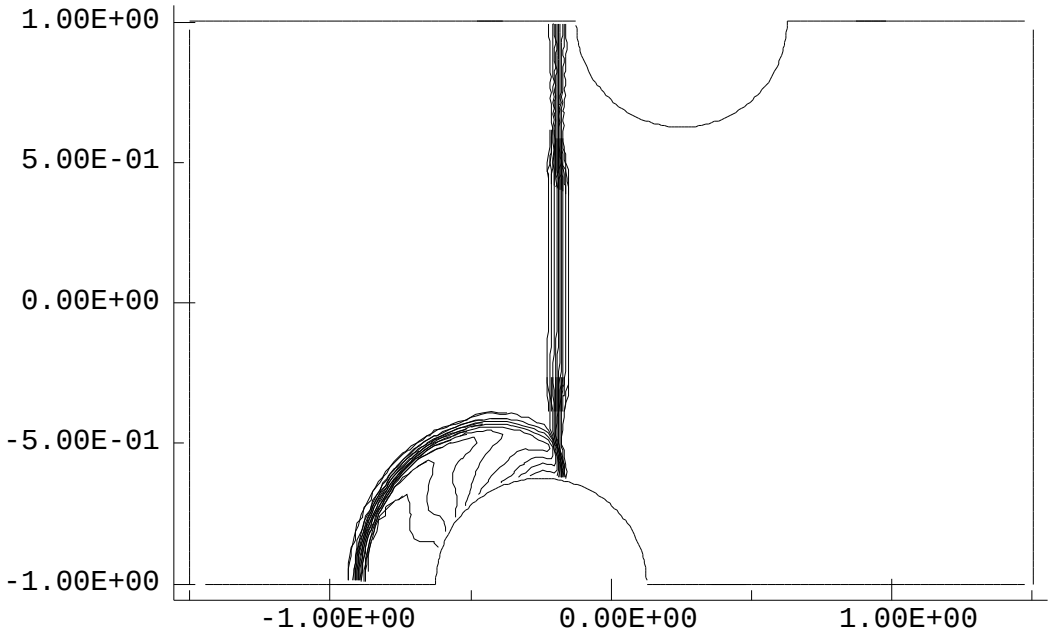


Figure 5: Fluid density at time 0.4.

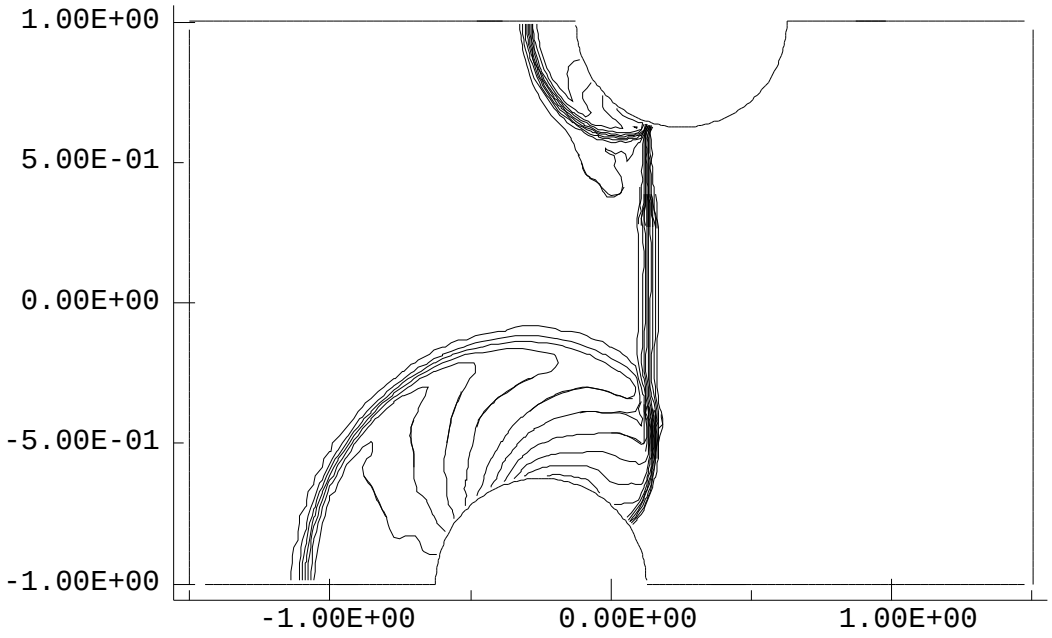


Figure 6: Fluid density at time 0.6.

lo= 0.98E+00 hi= 0.21E+01 inc= 0.56E-01

Figure 7: Fluid density at time 0.8.

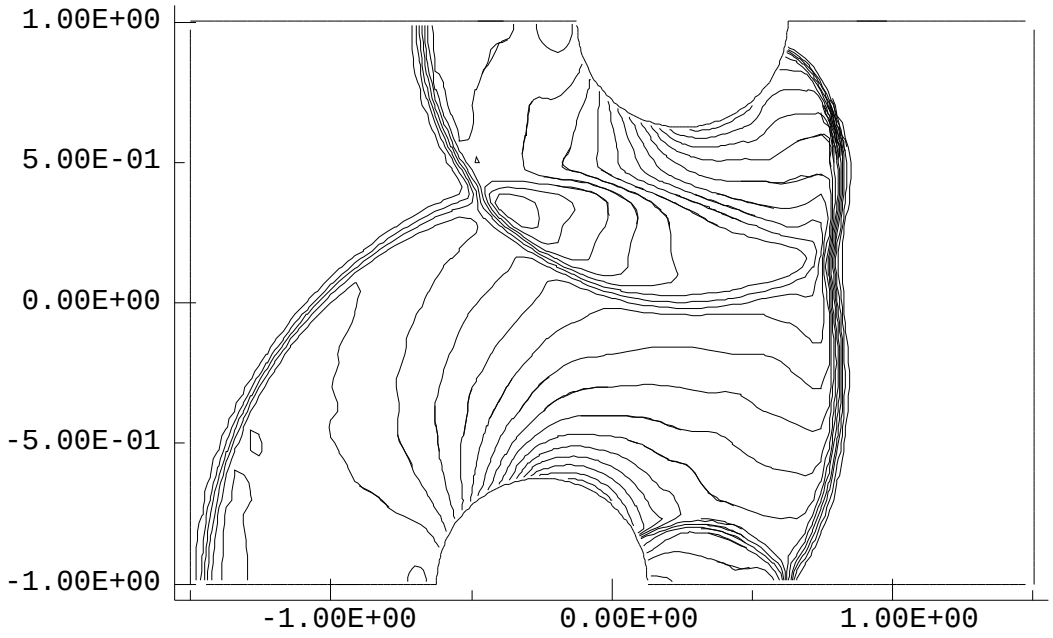
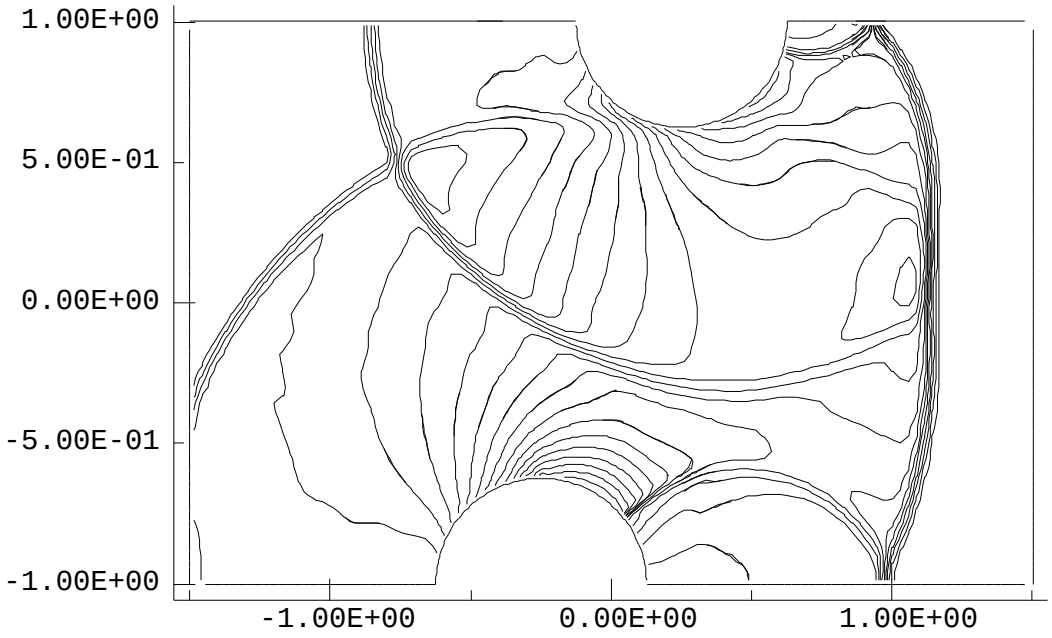


Figure 8: Fluid density at time 1.0.



lo= 0.93E+00 hi= 0.22E+01 inc= 0.62E-01

Figure 9: Fluid density at time 1.2.

References

[1] J. Bell, M. Berger, J. Saltzman, and M. Welcome, *Three dimensional adaptive mesh refinement for hyperbolic conservation laws*, LLNL unclassified report UCRL-JC-108794, Lawrence Livermore National Laboratory, 1991.

[2] J. Bell, P. Colella, and J. Trangenstein, *Higher order Godunov methods for general systems of hyperbolic conservations laws*, J. Comp. Phys., 82 (1989), pp. 362–397.

[3] J. A. Benek, P. G. Buning, and J. L. Steger, *A 3-d Chimera grid embedding technique*, in Proceedings of the 7th AIAA Computational Fluid Dynamics Conference, Cincinnati, AIAA, 1985, pp. 322–331.

[4] M. J. Berger, *On conservation at grid interfaces*, SIAM J. of Numer. Anal., 24 (1987), pp. 967–984.

[5] ——, *Private communiction*, 1993.

[6] M. J. Berger, K. D. Brislawn, and J. S. Saltzman, *Conservative interpolation on composite overlapping grids*, LANL unclassified report, Los Alamos National Laboratory, 1993.

[7] M. J. Berger and P. Colella, *Local adaptive mesh refinement for shock hydrodynamics*, J. Comp. Phys., 82 (1989), pp. 64–84.

[8] K. D. Brislawn, D. L. Brown, G. Chesshire, and J. S. Saltzman, *Adaptive composite overlapping grids for hyperbolic conservation laws*, LANL unclassified report, Los Alamos National Laboratory, 1993.

[9] D. Brown, G. Chesshire, W. Henshaw, and H. Kreiss, *On composite overlapping grids*, in Proceedings of the Seventh International Conference on Finite Element Methods in Flow Problems, 1989.

[10] D. L. Brown, G. Chesshire, and W. D. Henshaw, *An explanation of the CMPGRD composite grid data structure*, IBM research report, IBM Research Division, Yorktown Heights, NY, 1990.

[11] ——, *Getting started with CMPGRD, introductory user's guide and reference manual*, LANL unclassified report LA-UR-90-3729, Los Alamos National Laboratory, 1990.

[12] P. G. Buning, I. T. Chiu, S. Obayashi, Y. M. Rizk, and J. L. Steger, *Numerical simulation of the integrated space shuttle vehicle in ascent*, AIAA paper 88-4359-CP, AIAA, 1988.

[13] G. Chesshire and W. D. Henshaw, *The DSK package, a data structure for efficient fortran array storage, (reference guide for the DSK package)*, IBM Research Report RC 14353, IBM Research Division, Yorktown Heights, NY, 1988.

[14] ——, *Composite overlapping meshes for the solution of partial differential equations*, J. Comp. Phys., 90 (1990), pp. 1–64.

[15] ——, *Conservation on composite overlapping grids*, IBM Research Report RC 16531, IBM Research Division, Yorktown Heights, NY, 1991. To appear in SIAM J. Sci. Comput.

[16] G. S. Chesshire, *Composite Grid Construction and Applications*, PhD thesis, California Institute of Technology, 1986.

[17] P. Colella, *Multidimensional upwind methods for hyperbolic conservation laws*, J. Comp. Phys., 87 (1990), pp. 171–200.

[18] R. Courant and K. Friedrichs, *Supersonic Flow and Shock Waves*, Springer Verlag, New York, 1999.

[19] F. C. Dougherty and J. Kuan, *Transonic store separation using a three-dimensional Chimera grid scheme*, AIAA paper 89-0637, AIAA, 1989.

[20] A. Harten, P. Lax, and B. van Leer, *On upstream differencing and Godunov-type schemes for hyperbolic conservation laws*, SIAM Review, 25 (1983), pp. 35–61.

[21] W. D. Henshaw, *A fourth-order accurate method for the incompressible Navier-Stokes equations on overlapping grids*, IBM Research Report RC 18609, IBM Research Division, Yorktown Heights, NY, 1992. submitted to J. Comp. Phys.

[22] W. D. Henshaw, G. Chesshire, and M. Henderson, *On constructing three dimensional overlapping grids with CMPGRD*, in Software systems for surface modelling and grid generation, R. E. Smith, ed., NASA Conference Publication 3143, 1992, pp. 415–434. Proceedings of a workshop held at Langley Research Center, April 28-30, 1992.

[23] C. Hirsch, *Numerical Computation of Internal and External Flows*, vol. 1 and 2, John Wiley & Sons, New York, 1990.

[24] B. Kreiss, *Construction of a curvilinear grid*, SIAM J. Sci. Stat. Comput., 4 (1983), pp. 270–279.

[25] R. Leveque, *Numerical Methods for Conservation Laws*, Birkhauser Verlag, Boston, 1990.

[26] B. Müller and A. Rizzi, *Runge-Kutta finite-volume simulation of laminar transonic flow over a NACA0012 airfoil using the Navier-Stokes equations*, Technical Report FFA TN 1986-60, Aeronautical Research Institute of Sweden, 1986.

[27] K. Pao and J. Saltzman, *A comparison of approximate flux functions for 1-D gas dynamics*, LANL Unclassified Report LA-UR-90-3290, Los Alamos National Laboratory, 1990.

[28] E. Pärt-Enaander and B. Sjögreen, *Conservative and non-conservative interpolation between overlapping grids for finite volume solutions of hyperbolic equations*, technical report, CERFACS, 1993. To appear in Computers and Fluids.

[29] N. A. Petersson, *Computing gravity waves on water by using moving composite overlapping grids*, CAM Report 92-01, UCLA, 1992.

[30] N. A. Petersson and J. F. Malmliden, *Computing the flow around a submerged body using composite grids*, J. Comput. Phys., 105 (1993), pp. 47–57.

[31] E. G. Puckett and J. S. Saltzman, *A 3D adaptive mesh refinement algorithm for multimaterial gas dynamics*, Physica D, 60 (1992), pp. 84–93.

[32] A. Rizzi, *Numerical implementation of solid-body boundary conditions for the Euler equations*, ZAMM, 58 (1978), pp. T301 – T304.

[33] J. Saltzman, *An unsplit 3-D upwind method for hyperbolic conservation laws*, LANL Unclassified Report LA-UR-89-3442, Los Alamos National Laboratory, 1989. submitted to J. Comp. Physics.

[34] G. Starius, *On composite mesh difference methods for hyperbolic differential equations*, Numer. Math., 35 (1980), pp. 241–255.

[35] J. L. Steger and J. A. Benek, *On the use of composite grid schemes in computational aerodynamics*, Computer Methods in Applied Mechanics and Engineering, 64 (1987), pp. 301–320.